Task 02: JavaScript Mini Project

Internship Domain: Frontend Development

Intern Name: Manu Kumar

Task Number: 02

Task Title: JavaScript Mini Project

Submission Date: February 12, 2026

(A) Project Link:

- **GitHub Repository:** https://github.com/HADESOO7/Pro-Calculator/tree/main
- **Live Hosted Link:** https://hadesoo7.github.io/Pro-Calculator/

(B) Project Overview

Objective: To develop a fully functional, mobile-first scientific calculator that mimics the aesthetic and responsiveness of modern iOS applications. The project emphasizes complex logic handling (scientific functions) and dynamic DOM manipulation (history drawer).

Key Features:

- **Scientific Operations:** Supports Trigonometry (Sin, Cos, Tan), Logarithms, and Exponents.
- **Smart Evaluation:** Handles Degree/Radian switching and complex expression parsing.
- **History Management:** A slide-up drawer that stores past calculations for quick reference.
- **UI/UX:** Neumorphic/Dark-mode design with haptic-like visual feedback.

(C) Implementation Details & Code Snippets

1. CSS Styling (Theming & Layout): This snippet demonstrates your use of CSS Variables for easy theming and the specific styling to make the calculator look like a modern iOS app.

```css
:root {
    --display-bg: #111;
    --btn-num: #333333;
    --btn-op: #ff9f0a; /* iOS Orange */
    --btn-top: #a5a5a5; /* iOS Light Grey */
    --btn-sci: #212121; /* Darker grey for scientific */
    --text-white: #fff;
    --text-black: #000;
}

body {
    margin: 0;
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif;
    background-color: #1c1c1c;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    overflow: hidden;
}

/* Phone Container */
.calculator {
    width: 375px;
    height: 812px; /* iPhone X Aspect Ratio */
    background-color: var(--bg);
    border-radius: 40px;
    box-shadow: 0 0 50px rgba(0,0,0,0.8);
    display: flex;
    flex-direction: column;
    position: relative;
    overflow: hidden;
    border: 4px solid #333;
}

/* --- DISPLAY AREA --- */
.display {
    flex: 1;
    padding: 30px 25px;
    display: flex;
    flex-direction: column;
    justify-content: flex-end;
    align-items: flex-end;
    text-align: right;
}
```

2. Core Logic (JavaScript): This is the most important part. It shows to handle scientific operations, clean up the input (Regex), and manage errors using try-catch.

```
330     function calculate() {
331         try {
332             let evalStr = currentExp;
333 
334             // 1. Handle Power operator (^) -> Replace with Math.pow logic or **
335             // We replace x^y with x**y
336             evalStr = evalStr.replace(/\^/g, '**');
337 
338             // 2. Handle Implied Multiplication (e.g., 2sin(30) -> 2*sin(30))
339             // Insert * between number and function/parenthesis
340             evalStr = evalStr.replace(/(\d)(sin|cos|tan|log|ln|sqrt|\()/g, '$1*$2');
341 
342             // 3. Close parentheses automatically if missing
343             const openParens = (evalStr.match(/\(/g) || []).length;
344             const closeParens = (evalStr.match(/\)/g) || []).length;
345             if (openParens > closeParens) {
346                 evalStr += ")".repeat(openParens - closeParens);
347             }
348 
349             // 4. Evaluate using the helper functions above
350             let result = eval(evalStr);
351 
352             // 5. Formatting
353             if (!isFinite(result)) {
354                 result = "Error";
355             } else {
356                 // Remove floating point errors (e.g. 0.0000000004)
357                 result = parseFloat(result.toFixed(8)).toString();
358             }
359 
360             // 6. Update UI
361             displayExp.innerText = currentExp + " ="; // Move equation to top
362             displayRes.innerText = result; // Show answer
363 
364             // 7. Save to History
365             addToHistory(currentExp, result);
366 
367             // 8. Prepare for next operation
368             currentExp = result;
369 
370         } catch (e) {
371             displayRes.innerText = "Error";
372             setTimeout(() => { clearAll(); }, 1500);
373         }
374     }
```

3. History Feature (DOM Manipulation): This snippet shows that you know how to create HTML elements dynamically using JavaScript (not just static HTML)

```
    // --- HISTORY ---
    function toggleHistory() {
        document.getElementById('historyDrawer').classList.toggle('open');
    }

    function addToHistory(exp, res) {
        if(res === "Error") return;

        const div = document.createElement('div');
        div.className = 'history-item';
        div.innerHTML = `<div class="hist-exp">${exp} =</div><div class="hist-res">${res}</div>`;

        // Tap history to reuse result
        div.onclick = () => {
            currentExp += res;
            updateDisplay();
            toggleHistory();
        };

        historyList.prepend(div);
    }

    function clearHistory() {
        historyList.innerHTML = '';
        toggleHistory();
    }
</script>

</body>
</html>
```

(D) Execution Commands

To run this project locally:

1. **Clone the repository:** git clone https://github.com/HADESOO7/Pro-Calculator.git
2. **Launch:** Open index.html in any standard web browser.

(E) Interface Screenshots

Figure 1: Main Calculator.

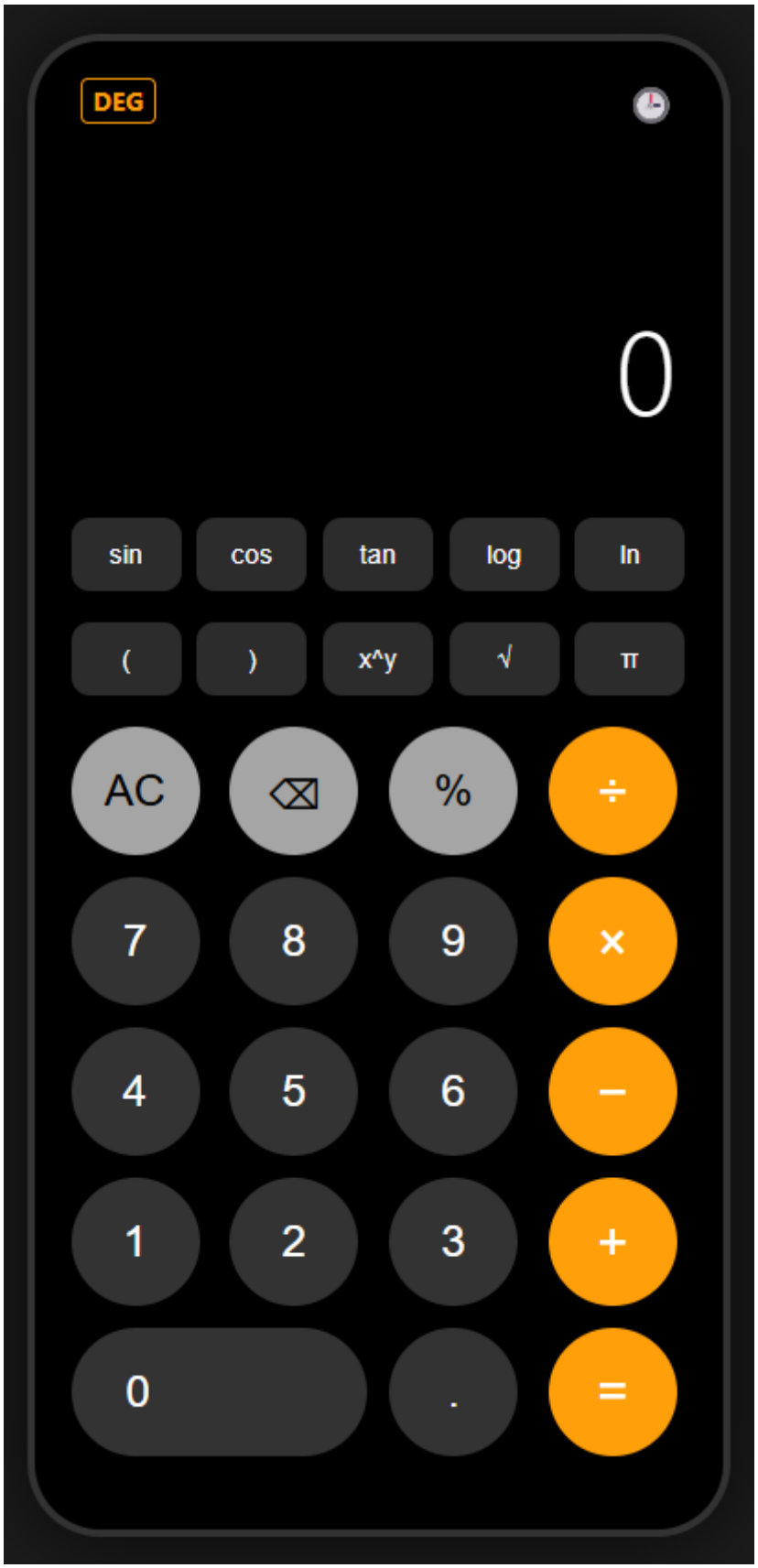

Figure 2: Scientific Mode & Degree/Radian Toggle

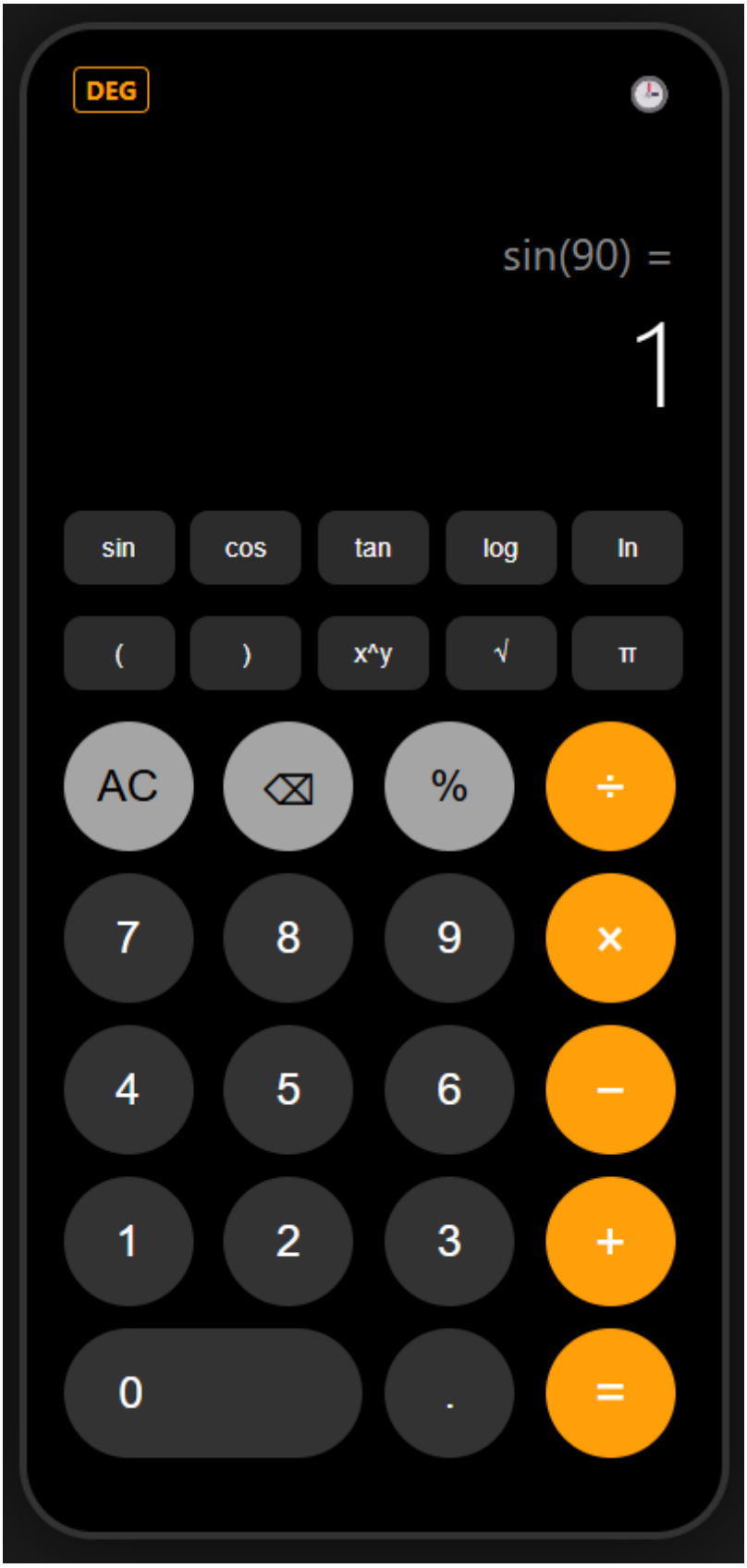

Figure 3: History Drawer (Slide-up UI)